Computing requirements for long-term research

Keith Briggs

Keith.Briggs@bt.com

more.btexact.com/people/briggsk2

CRG meeting 2004 June 25 1500

sodium:tex/computing2004jun25.tex

TYPESET 2004 JUNE 25 10:41 IN PDFLATEX ON A LINUX SYSTEM

Introduction

- What computing do we do?
  - *simulation*
  - *computer algebra*
  - *combinatorics and graph theory*
  - *solving ODEs*
  - *MC and stochastic processes*
  - *computational geometry*
  - *document and diagram production*

- What is our philosophy?
  - *experimental mathematics*
  - *open-mindedness - try every idea - flexibility*
  - *attack the problem from every possible angle*

- ∴ we need an open platform

Computers

- hierarchy of storage devices: fast & small to slow & big. . .
  - *registers*
  - *on-chip cache*
  - *on-board cache*
  - *RAM*
  - *harddisk*
  - *CD*
  - *tape*
  - *network*
  - . . .
- the challenge for us is to write programs that use these resources optimally

Unix philosophy

- goes back about 30 years to ATT and Berkeley
- a lot has been reinvented by MS, but not so well
- small is good, monolithic is bad
- make each program do one job only, but well
- make it easy to use this programs together
- pipe: `prog1 | prog2 | prog3 | ...`
- everything is open-source and uses open standards

Computing

★ . . . is not just using a monolithic package!

★ there are many languages, but different ones are optimal for different purposes

- *python for text processing, front-ending and glue*
- *C and C++ for heavy number-crunching*
- *etc.. . .*

★ time-space tradeoff -- e.g. sparse matrix techniques

Mathematical software

- ⋆ BLAS, lapack for numerical linear algebra
- ⋆ GSL for most other numerical functions
- ⋆ octave, scilab for matlab
- ⋆ GAP for group theory
- ⋆ R for statistics
- ⋆ Singular, Macaulay2 for polynomials and singularity theory
- ⋆ maxima for computer algebra
- ⋆ lp_solve for linear programming
- ⋆ nauty, graphviz for graph theory
- ⋆ 1000s of others. . .

Example: GAP

Analyze Cayley graph of small groups. . .

```
f:=FreeGroup(m);
fgens:=GeneratorsOfGroup(f);
ngrp:=NumberSmallGroups(n);
for k in [1..ngrp] do
  g:=SmallGroup(n,k);
  q:=GQuotients(f,g);
  l:=List(q,x->Image(x,fgens));
  for i in [1..Length(l)] do
    c:=CayleyGraph(g,l[i]);
    a:=Adjacency(c,1); d:=0;
    Print(c,a);
  od;
od;
```

Example: singular

Derive a Runge-Kutta third-order method:

$$\begin{aligned} b_3\ a_{32}\ c_2 &= 1/6 \\ b_1+b_2+b_3 &= 1 \\ b_2\ c_2+b_3c_3 &= 1/2 \\ b_2\ c_2^2+b_3\ c_3^2 &= 1/3 \end{aligned}$$

```
ring r=0,(b1,b2,b3,c2,c3,a32);
ideal i;
i=
  3*c2-1, // Heun
  3*c3-2, // Heun
  b1+b2+b3-1,               // 1.11.a
  2*(b2*c2+b3*c3)-1,        // 1.11.b
  3*(b2*c2^2+b3*c3^2)-1,    // 1.11.c
  6*b3*a32*c2-1             // 1.11.d
;
option(redSB);
```

```
ideal j=groebner(i);
list k=triangMH(j,2);
triang_solve(k,20);
rlist;
```

Output:

```
0.25
0
0.75
0.33333333333333333333
0.66666666666666666667
0.66666666666666666667
```

Example: nauty

How many non-isomorphic connected graphs of 14 nodes, regular of degree 4 are there?

```
geng -c -u 14 -d4 -D4
>Z 88168 graphs generated in 46.41 sec
```

How many non-isomorphic connected triangle-free and 4-cycle-free graphs of 14 nodes are there?

```
geng -c -u  -t -f 14
>A geng -ctfd1D13 n=14 e=13-23
>Z 275480 graphs generated in 2.32 sec
```

Example: how to make a long simulation reliable

- consider the simulation as a mapping from an input file to an output file
- let the mapping be one-to-one on lines
- the algorithm (© KMB 2004): repeat these steps indefinitely:
  - *count the lines in the output file*
  - *read the same number of lines from the input file*
  - *read the next line from the input file*
  - *using this line as input, compute the output*
  - *open the output file, append the new output line, and close the output file*
- note that the files are kept closed most of the time, in case of power cuts, system crashes, reboots etc.

Why I need a big, fast computer

★ 2 or 4GB RAM, 3GHz CPU (or 2 CPU)

- ▷ *we can share one machine and access it by remote login*
- ▷ *brute-force vs. smart computing*
- ▷ *intermediate expression swell in computer algebra*
- ▷ *we can do experiments not otherwise feasible, e.g. very large graphs*
- ▷ *we can solve very big optimization problems*
- ▷ *we can do better combinatorics - e.g., I have been able to enumerate connected graphs only up to 35 nodes on my 1G RAM machine*
- ▷ *we can better MC stochastic simulations - the more trials, the better the statistics*
- ▷ *I can do better diophantine approximation work*